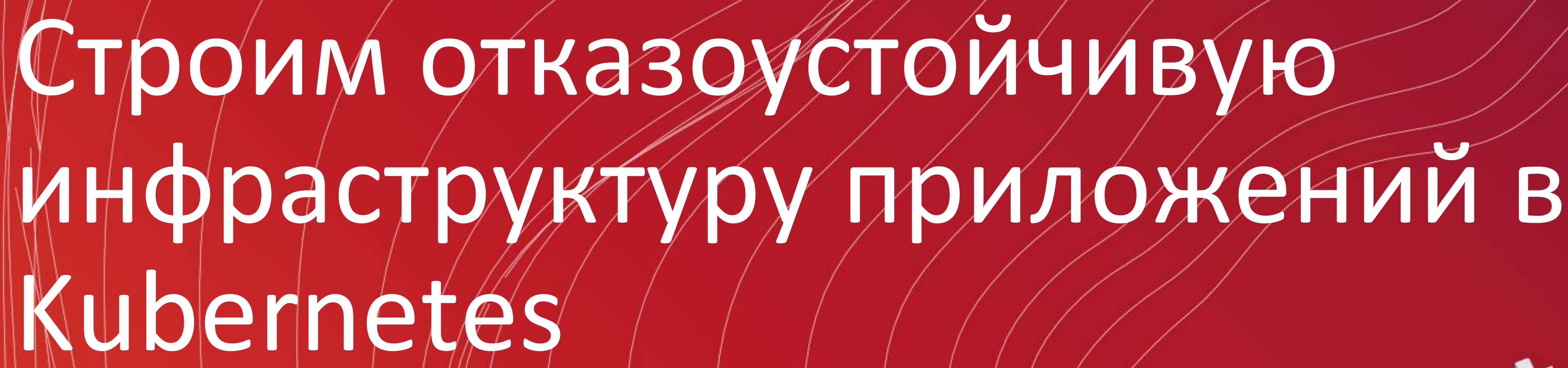


Строим отказоустойчивую инфраструктуру приложений в Kubernetes

Олег Вознесенский







Олег Вознесенский
Руководитель разработки отдела развития инфраструктуры для анализа данных



Газпромбанк — входит в тройку крупнейших универсальных банков России и занимает третье место в списке банков Центральной и Восточной Европы по размеру собственного капитала.

Мы не просто банкиры: мы создаем искусственный интеллект, разрабатываем квантовые компьютеры, придумываем новые цифровые продукты, поддерживаем профессиональное комьюнити, не забывая при этом, что в центре инноваций — человек.

Поэтому мы уделяем особую роль развитию. Мы создаем IT-сообщества и с помощью школы спикеров помогаем коллегам делиться успехами. Регулярно проводим митапы и даем возможность повышать квалификацию во внутренних школах разработки.

ПАТТЕРНЫ ОТКАЗОУСТОЙЧИВОСТИ ПРИЛОЖЕНИЙ

в Kubernetes



Что такое отказоустойчивость?



Отказоустойчивость – способность системы сохранять свою работоспособность после отказа одной или нескольких её составных частей.

Что такое отказоустойчивость?



Отказоустойчивость – способность системы сохранять свою работоспособность после отказа одной или нескольких её составных частей.

Отказоустойчивая архитектура — это способ построения отказоустойчивых систем, которые сохраняют работоспособность (возможно, с понижением эффективности) при отказах элементов.

Что такое отказоустойчивость?



Отказоустойчивость – способность системы сохранять свою работоспособность после отказа одной или нескольких её составных частей.

Отказоустойчивая архитектура — это способ построения отказоустойчивых систем, которые сохраняют работоспособность (возможно, с понижением эффективности) при отказах элементов.

Application level - 12 factor app
System level - kubernetes
Hardware level - redundancy

Application level



Паттерны отказоустойчивой архитектуры

Александр Кривощёков
Яндекс Go

Yandex for developers*//>

https://youtu.be/YlXJMCdssAI

Что такое отказоустойчивость?



Отказоустойчивость – способность системы сохранять свою работоспособность после отказа одной или нескольких её составных частей.

Отказоустойчивая архитектура — это способ построения отказоустойчивых систем, которые сохраняют работоспособность (возможно, с понижением эффективности) при отказах элементов.

Application level - 12 factor app
System level - kubernetes
Hardware level - redundancy

Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit breaker
- Rate limits
- Rollout

Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit breaker
- Rate limits
- Rollout

Watchdog



SERVER

SERVER

Watchdog





SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Watchdog



CLIENT

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Watchdog



CLIENT
SERVER
Service #1
Service #2
SERVER
Service #1
Service #2

Watchdog



CLIENT

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Watchdog



CLIENT

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Watchdog

Watchdog



CLIENT
SERVER
Service #1
Service #2
SERVER
Service #1
Service #2
Watchdog

Watchdog



SERVER

Service #1

Service #2

CLIENT

Watchdog

SERVER

Service #1

Service #2

Watchdog



SERVER
Service #1
Service #2
CLIENT
Watchdog
SERVER
Service #1
Service #2

Watchdog



SERVER
Service #1
Service #2
CLIENT
Watchdog
SERVER
Service #1
Service #2

Watchdog



CLIENT

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Watchdog

Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Rate limits
- Circuit breaker
- Rollout

Health check





SERVER

Service #1

Service #2

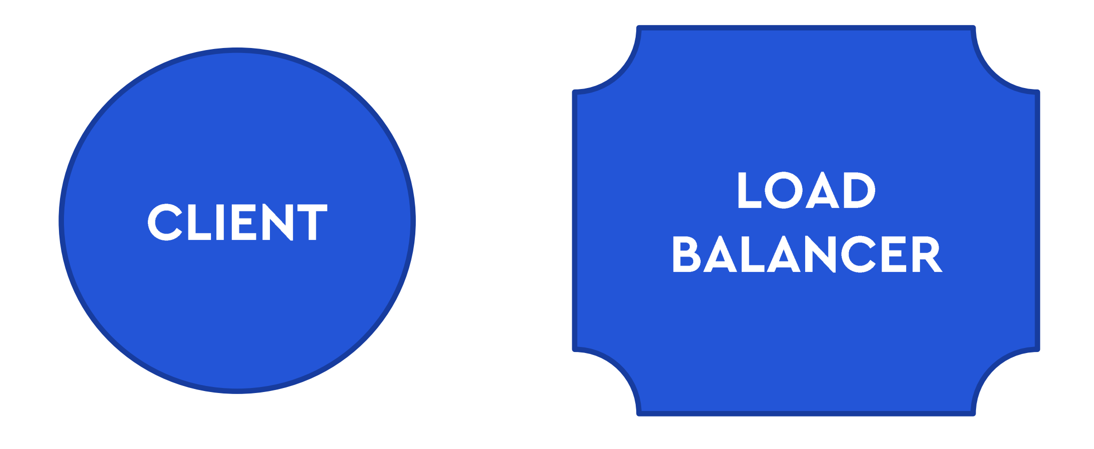
SERVER

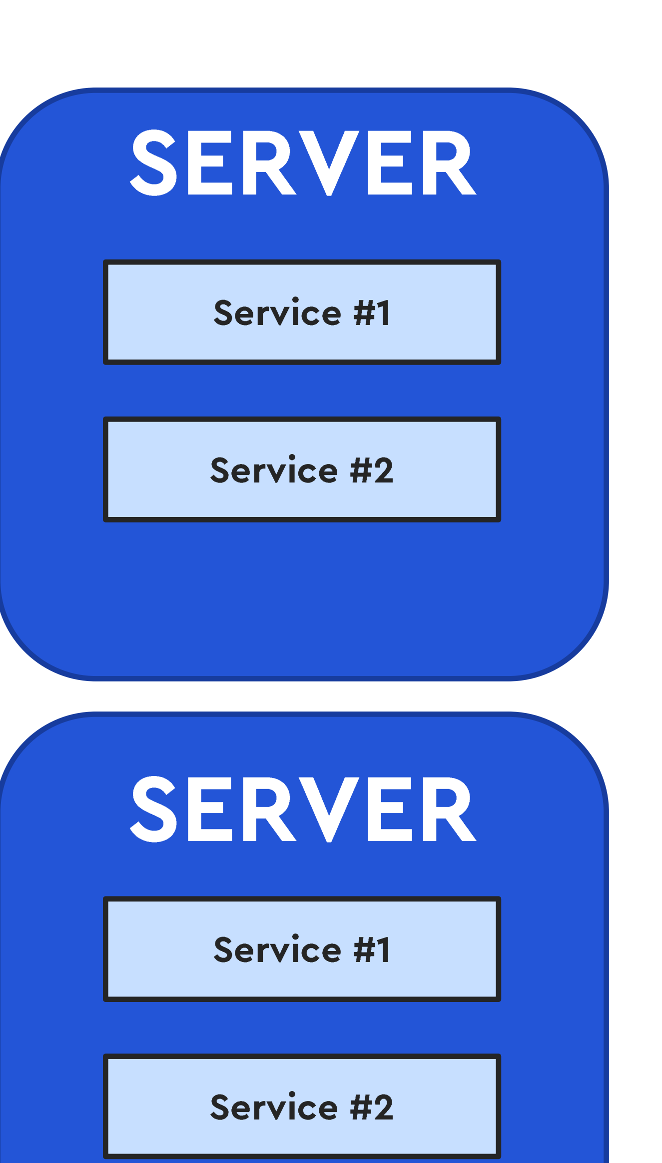
Service #1

Service #2

Health check







Health check



CLIENT

LOAD
BALANCER

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Health check



CLIENT

LOAD BALANCER

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Health check



CLIENT

LOAD BALANCER

SERVER

Service #1

Service #2

SERVER

Service #1

Service #2

Health check



Health check



Kubernetes probes



- Watchdog
- Health check

Kubernetes probes



- Watchdog = Liveness probe
- Health check = Readiness probe

Watchdog + healthcheck



Kubernetes probes



Kubernetes probes – pod lifecycle



- Liveness probe
- Readiness probe

Независимые!

- initialDelaySeconds
- periodSeconds
- timeoutSeconds
- successThreshold
- failureThreshold



Kubernetes probes – pod lifecycle



- Liveness probe
- Readiness probe

Независимые!

- initialDelaySeconds
- periodSeconds
- timeoutSeconds
- successThreshold
- failureThreshold

Create pod

T

Kubernetes probes – pod lifecycle



- Liveness probe
- Readiness probe

Независимые!

Create pod

T

Kubernetes probes – pod lifecycle



- Startup probe
- Liveness probe
- Readiness probe

Create pod

Kubernetes probes – pod lifecycle



- Startup probe
- Liveness probe
- Readiness probe

Java
- JIT
- Liquibase migrations

Create pod

Kubernetes probes – pod lifecycle



- Startup probe — рестарт контейнера
- Liveness probe — рестарт контейнера
- Readiness probe — управление трафиком



Kubernetes service

- Service discovery - DNS record
- Load Balancing - endpoint

Kubernetes probes – pod lifecycle



- Startup probe
- Liveness probe
- Readiness probe

- exec
- grpc
- httpGet
- tcpSocket



ХОРОШАЯ ПРОБА

Kubernetes probes – хорошая проба



- Startup probe
- Liveness probe
- Readiness probe

- Проверка бизнес функционала приложения (осторожнее с зависимостями от внешних сервисов)
- Учёт особенностей приложения

Kubernetes probes – хорошая проба



CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

kubelet

Kubernetes probes – хорошая проба



CLIENT

service

POD1

Nginx

php-fpm

POD2

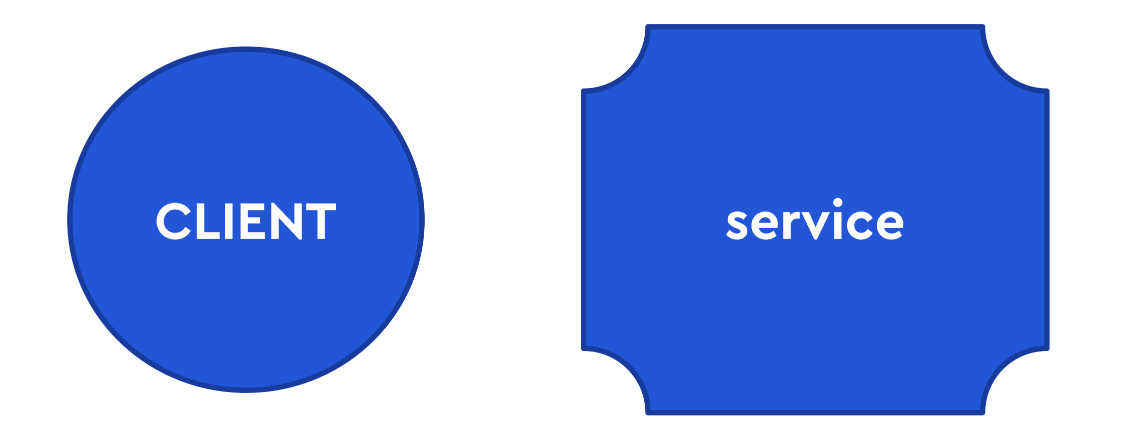
Nginx

php-fpm

kubelet

Kubernetes probes – хорошая проба – readiness





POD1

Nginx

php-fpm

kubelet

POD2

Nginx

php-fpm

Kubernetes probes – хорошая проба – liveness



php-fpm – pools

```
[site]
user = www
group = www
listen = 127.0.0.1:9001
pm = dynamic
pm.max_children = 20
pm.start_servers = 10
pm.min_spare_servers = 5
pm.max_spare_servers = 10
pm.process_idle_timeout = 10s

[healthcheck]
user = www
group = www
listen = 127.0.0.1:9002
pm = static
pm.max_children = 1
```

Kubernetes probes – хорошая проба – liveness



php-fpm – pools

```
[site]
user = www
group = www
listen = 127.0.0.1:9001
pm = dynamic
pm.max_children = 20
pm.start_servers = 10
pm.min_spare_servers = 5
pm.max_spare_servers = 10
pm.process_idle_timeout = 10s

[healthcheck]
user = www
group = www
listen = 127.0.0.1:9002
pm = static
pm.max_children = 1
```

Kubernetes probes – хорошая проба – liveness



php-fpm – pools

```
[site]
user = www
group = www
listen = 127.0.0.1:9001
pm = dynamic
pm.max_children = 20
pm.start_servers = 10
pm.min_spare_servers = 5
pm.max_spare_servers = 10
pm.process_idle_timeout = 10s

[healthcheck]
user = www
group = www
listen = 127.0.0.1:9002
pm = static
pm.max_children = 1
```

Kubernetes probes – хорошая проба – liveness



php-fpm – pools

```
[site]
user = www
group = www
listen = 127.0.0.1:9001
pm = dynamic
pm.max_children = 20
pm.start_servers = 10
pm.min_spare_servers = 5
pm.max_spare_servers = 10
pm.process_idle_timeout = 10s

[healthcheck]
user = www
group = www
listen = 127.0.0.1:9002
pm = static
pm.max_children = 1
```

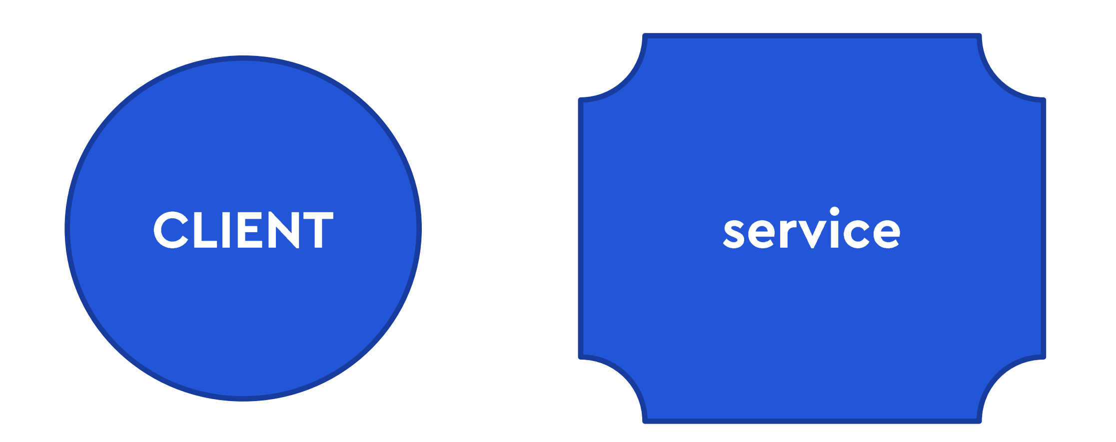
Паттерны отказоустойчивости

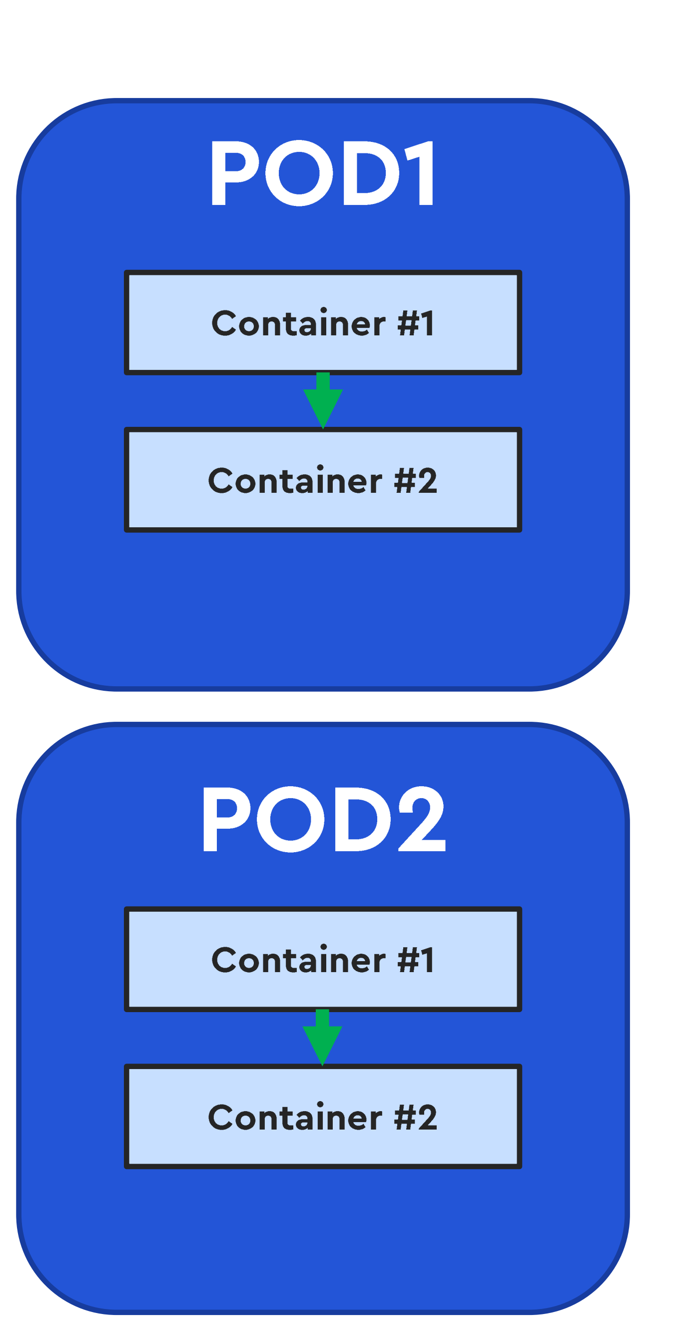


- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit breaker
- Rate limits
- Rollout

Retry/Timeout

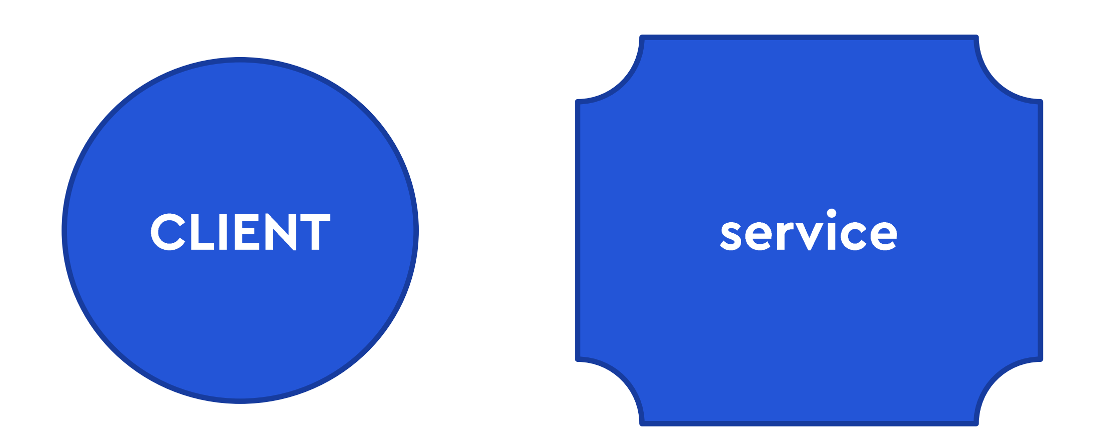


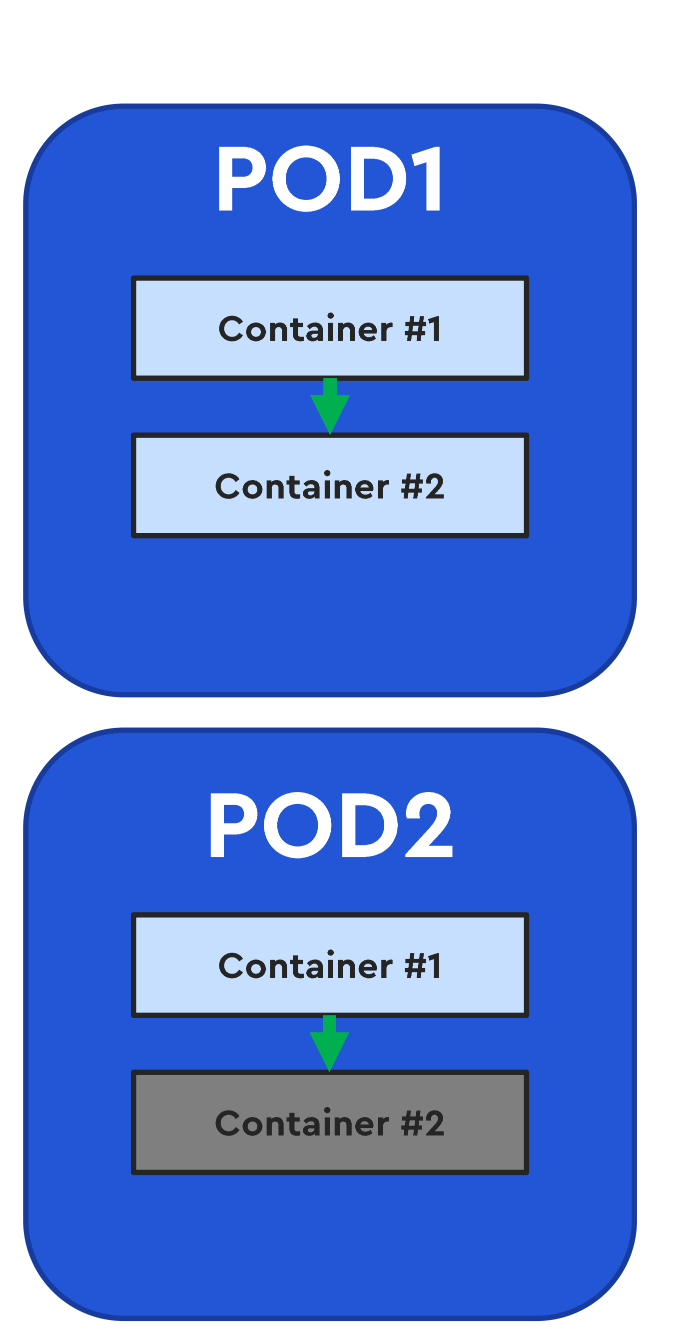




Retry/Timeout







Retry/Timeout



POD1

Container #1

Container #2

CLIENT

service

POD2

Container #1

Container #2

Retry/Timeout



CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

- HTTP 5xx, 429
- Долгий ответ

Retry/Timeout



POD1

Container #1

Container #2

CLIENT

service

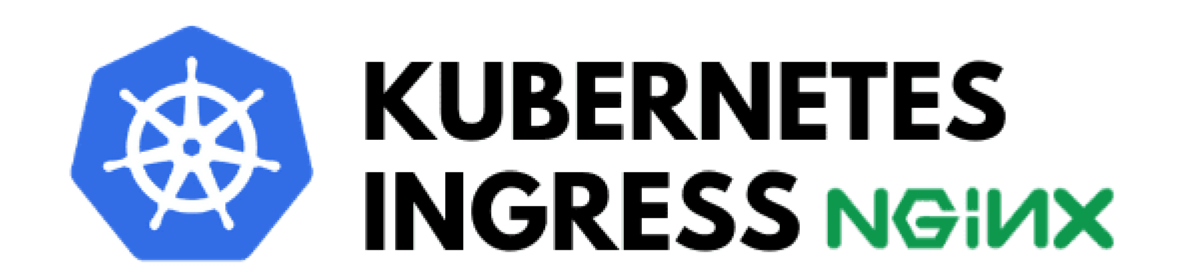
POD2

Container #1

Container #2

Retry/Timeout







Ingress

Service mesh

Retry/Timeout - ingress



Retry/Timeout - ingress



CLIENT

Ingress

service

POD1

Nginx

php-fpm

POD2

Nginx

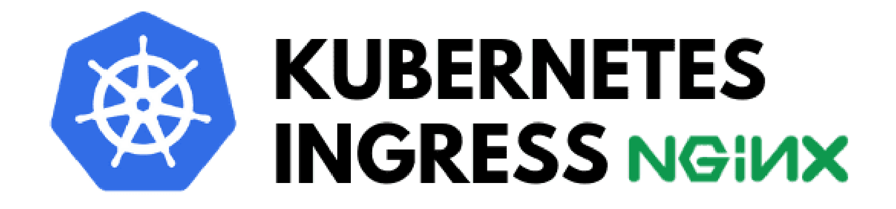
php-fpm

Retry/Timeout - ingress nginx



Retry

- proxy-next-upstream — условие повтора
- proxy-next-upstream-timeout — время
- proxy-next-upstream-tries — число



Retry/Timeout - ingress nginx



Retry

- proxy-next-upstream — условие повтора
- proxy-next-upstream-timeout — время
- proxy-next-upstream-tries — число

error | timeout | invalid_header |
http_500 | http_502 | http_503 | http_504 |
http_403 | http_404 | http_429 |
non_idempotent | off ...;

Retry/Timeout - ingress nginx



Retry

• proxy-next-upstream	условие повтора
• proxy-next-upstream-timeout	время
• proxy-next-upstream-tries	число

Timeout/Deadline

• proxy-connect-timeout	время
• proxy-read-timeout	время
• proxy-send-timeout	время

Retry/Timeout - ingress nginx



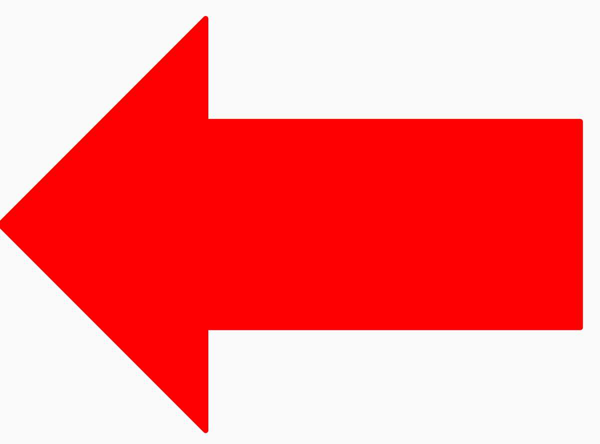
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ingress-nginx-controller
  namespace: ingress-nginx
data:
  proxy-next-upstream: "error timeout http_500"
  proxy-next-upstream-timeout: "10"
  proxy-next-upstream-tries: "5"
  proxy-connect-timeout: "1"
  proxy-read-timeout: "5"
  proxy-send-timeout: "2"
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/

Retry/Timeout - ingress nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-snippet
  annotations:
    nginx.ingress.kubernetes.io/proxy-next-upstream: "error timeout http_500"
    nginx.ingress.kubernetes.io/proxy-next-upstream-timeout: "10"
    nginx.ingress.kubernetes.io/proxy-next-upstream-tries: "5"
    nginx.ingress.kubernetes.io/proxy-connect-timeout: "1"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "5"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "2"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```



https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#custom-timeouts

Retry/Timeout - ingress nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-snippet
  annotations:
    nginx.ingress.kubernetes.io/configuration-snippet: |
      proxy_next_upstream: "error timeout http_500"
      proxy_next_upstream_timeout: "10"
      proxy_next_upstream_tries: "5"
      proxy_connect_timeout: "1"
      proxy_read_timeout: "5"
      proxy_send_timeout: "2"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/examples/customization/configuration-snippets/

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
  GET, HEAD, OPTIONS, TRACE, PUT, DELETE

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
  GET, HEAD, OPTIONS, TRACE, PUT, DELETE

retry-non-idempotent ОСТОРОЖНО!!!
POST, LOCK, PATCH

Ключи идемпотентности

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/#retry-non-idempotent

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
- Настройка разумной политики retry/timeout

proxy-next-upstream-timeout
proxy-next-upstream-tries

proxy-connect-timeout
proxy-read-timeout
proxy-send-timeout

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
- Настройка разумной политики retry/timeout

proxy-next-upstream-timeout
proxy-next-upstream-tries

proxy-connect-timeout
proxy-read-timeout
proxy-send-timeout

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
- Настройка разумной политики retry/timeout

proxy-next-upstream-timeout
proxy-next-upstream-tries

proxy-connect-timeout
proxy-read-timeout
proxy-send-timeout

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
- Настройка разумной политики retry/timeout

proxy-next-upstream-timeout
proxy-next-upstream-tries

proxy-connect-timeout
proxy-read-timeout
proxy-send-timeout

Retry/Timeout на ingress nginx – особенности



- Только идемпотентные запросы!
- Настройка разумной политики retry/timeout
- clusterIp: none

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  clusterIP: none
  selector:
    app.kubernetes.io/name: myapp
  ports:
    - protocol: TCP
      port: 80
```

Retry/Timeout



Istio

Retry/Timeout – service mesh



POD1

Nginx

php-fpm

POD 3

Client APP

service

POD2

Nginx

php-fpm

Retry/Timeout – service mesh



POD 3

Client APP

Sidecar proxy

service

service config
for service mesh

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Service
mesh
control
plane

Retry/Timeout – service mesh



POD 3

Client APP

Sidecar proxy

service

service config for service mesh

POD1

Nginx

php-fpm

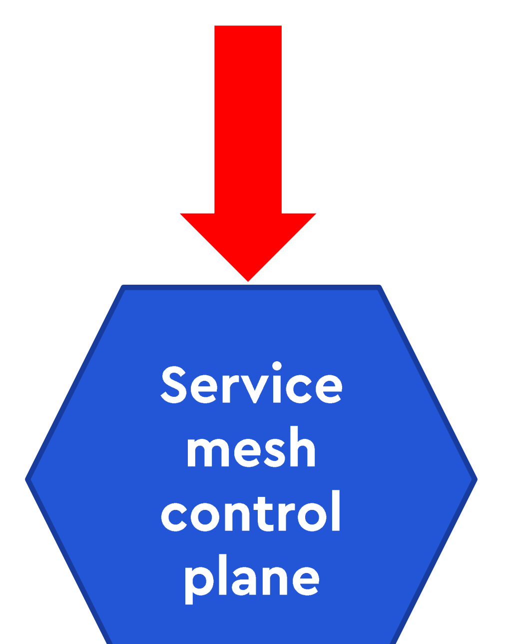
Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy



Retry/Timeout – service mesh



Retry/Timeout – service mesh



Retry/Timeout – service mesh



Sidecar



Retry/Timeout – service mesh



POD1

Nginx

php-fpm

Sidecar proxy



Client APP

Sidecar proxy

service

service config
for service mesh

POD2

Nginx

php-fpm

Sidecar proxy

Service
mesh
control
plane

Retry/Timeout – service mesh



POD 3

Client APP

Sidecar proxy

service

service config
for service mesh

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

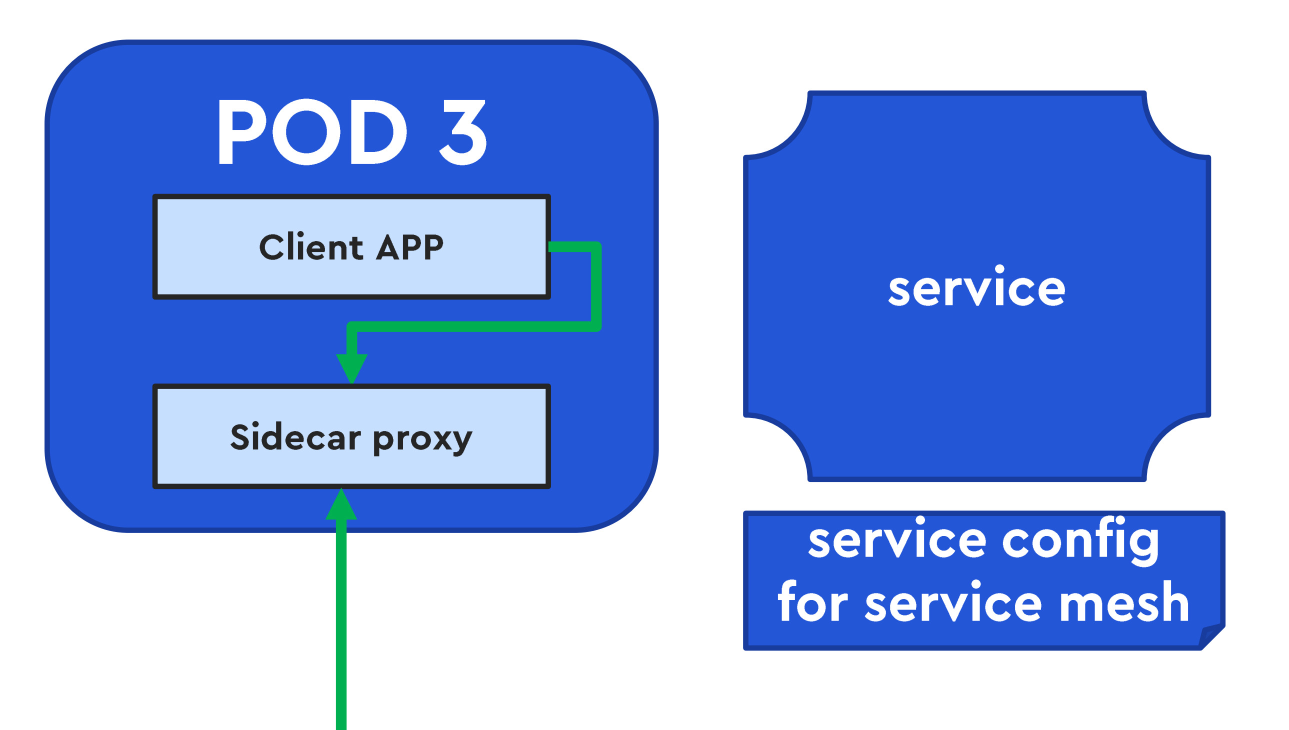
php-fpm

Sidecar proxy

Service
mesh
control
plane

Retry/Timeout – service mesh





Retry/Timeout – service mesh



POD1
Nginx
php-fpm
Sidecar proxy

POD 3
Client APP
Sidecar proxy

service

service config
for service mesh

POD2
Nginx
php-fpm
Sidecar proxy

Service
mesh
control
plane

Retry/Timeout – service mesh



Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
      - destination:
          host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
      - destination:
          host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
      - destination:
          host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
      - destination:
          host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
    - destination:
        host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Retry/Timeout - istio - VirtualService



```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: microservice-route
spec:
  hosts:
  - "*"
  http:
  - match:
    - method:
        regex: "GET|HEAD|TRACE"
    route:
      - destination:
          host: microservice.prod.svc.cluster.local
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: connect-failure,refused-stream,gateway-error,503
    timeout: 5s
```

Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit Breaker
- Rate limits
- Rollout

Circuit Breaker – service mesh



POD 3

Client APP

Sidecar proxy

service

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker – service mesh



POD 3

Client APP

Sidecar proxy

service

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker – service mesh



POD1
Nginx
php-fpm
Sidecar proxy

POD 3
Client APP
Sidecar proxy

service

Service mesh control plane

POD2
Nginx
php-fpm
Sidecar proxy

Circuit Breaker – service mesh



POD1

Nginx

php-fpm

Sidecar proxy

POD 3

Client APP

Sidecar proxy

service

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker – service mesh



Circuit Breaker – service mesh



POD1

Nginx

php-fpm

Sidecar proxy

POD 3

Client APP

Sidecar proxy

service

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker – service mesh



POD 3

Client APP

Sidecar proxy

service

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker – service mesh



POD 3

Client APP

Sidecar proxy

service

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Service mesh control plane

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

https://istio.io/latest/docs/reference/config/networking/destination-rule/#OutlierDetection

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Circuit Breaker - istio



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-prod
spec:
  host: microservice.prod.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        connectTimeout: 200ms
    outlierDetection:
      consecutive5xxErrors: 7
      interval: 10s
      baseEjectionTime: 5m
      maxEjectionPercent: 100
      minHealthPercent: 50
```

Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit Breaker
- Rate limits
- Rollout

Rate limits



CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

service

POD1

Container #1

Container #2

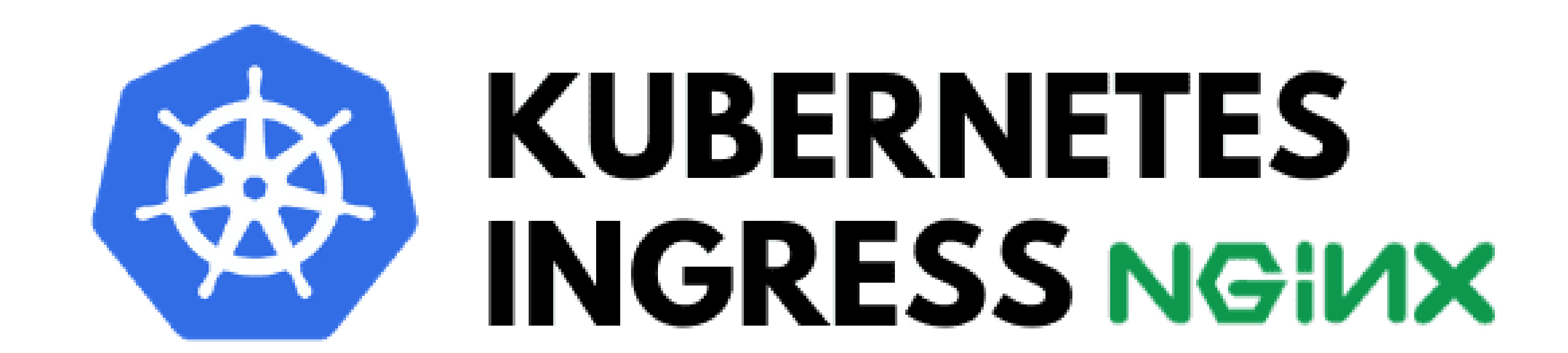
POD2

Container #1

Container #2

Rate limits - ingress





Rate limits - ingress



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

Ingress

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits - ingress



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

Ingress

Ingress

Ingress

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits - ingress-nginx - типы



- Rate Limiting (Local)
- Global Rate Limiting
  - lua-resty-global-throttle
  - memcached

Rate limits (Local) - ingress-nginx - типы



- Rate Limiting (Local)
- Global Rate Limiting
  - lua-resty-global-throttle
  - memcached

Rate limits (Local) - ingress-nginx - Local



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-example
  annotations:
    nginx.ingress.kubernetes.io/limit-connections: 100
    nginx.ingress.kubernetes.io/limit-rpm: 200
    nginx.ingress.kubernetes.io/limit-rps: 10
    nginx.ingress.kubernetes.io/limit-burst-multiplier: 2
    nginx.ingress.kubernetes.io/limit-whitelist: 8.8.8.8
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов
  nginx.ingress.kubernetes.io/limit-connections
  nginx.ingress.kubernetes.io/limit-rps
  nginx.ingress.kubernetes.io/limit-rpm

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов
  nginx.ingress.kubernetes.io/limit-connections
  nginx.ingress.kubernetes.io/limit-rps
  nginx.ingress.kubernetes.io/limit-rpm

limit-connections -> limit-rpm -> limit-rps

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов
  nginx.ingress.kubernetes.io/limit-connections
  nginx.ingress.kubernetes.io/limit-rps
  nginx.ingress.kubernetes.io/limit-rpm

limit-connections -> limit-rpm -> limit-rps

- Cloud load balancer/HAProxy

- HTTP Proxy

Rate limits (Local) - ingress-nginx - Local



- **Настройки лимитов**
  nginx.ingress.kubernetes.io/limit-connections
  nginx.ingress.kubernetes.io/limit-rps
  nginx.ingress.kubernetes.io/limit-rpm

limit-connections -> limit-rpm -> limit-rps

- **Cloud load balancer/HAProxy**

PROXY protocol!!!

use-proxy-protocol: “true”

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/#use-proxy-protocol

Rate limits (Local) - ingress-nginx - Local



- **Настройки лимитов**

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm

limit-connections -> limit-rpm -> limit-rps

- **Cloud load balancer**

PROXY protocol!!

- **HTTP Proxy**

X-Forwarded-For / use-forwarded-headers

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/#use-forwarded-headers

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm
nginx.ingress.kubernetes.io/limit-burst-multiplier

Rate limits (Local) - ingress-nginx - Local



- **Настройки лимитов**

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm
nginx.ingress.kubernetes.io/limit-burst-multiplier

Processed (rate=2, no burst)

Rate
Rejected
Processed

6
4,5
3
1,5
0

1 2 3 4 5 6 7 8 9 10 11

Time step

Processed (rate=2, burst=2)

Available burst tokens
Rejected
Processed (after delay)
Processed

6
5
4
3
2
1
0

1 2 3 4 5 6 7 8 9 10 11

Time step

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm
nginx.ingress.kubernetes.io/limit-burst-multiplier
nginx.ingress.kubernetes.io/limit-rate
nginx.ingress.kubernetes.io/limit-rate-after

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm
nginx.ingress.kubernetes.io/limit-burst-multiplier
nginx.ingress.kubernetes.io/limit-rate
nginx.ingress.kubernetes.io/limit-rate-after

nginx.ingress.kubernetes.io/proxy-buffering: "on"

Rate limits (Local) - ingress-nginx - Local



- Настройки лимитов

nginx.ingress.kubernetes.io/limit-connections
nginx.ingress.kubernetes.io/limit-rps
nginx.ingress.kubernetes.io/limit-rpm
nginx.ingress.kubernetes.io/limit-burst-multiplier
nginx.ingress.kubernetes.io/limit-rate
nginx.ingress.kubernetes.io/limit-rate-after
nginx.ingress.kubernetes.io/limit-whitelist:

Rate limits (Local) - ingress



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

Ingress

Ingress

Ingress

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Rate limits (Local) - ingress



CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT
CLIENT

Ingress

Ingress

Ingress

service

POD1

Container #1

Container #2

POD2

Container #1

Container #2

Global Rate limits - ingress-nginx



- Rate Limiting (Local)
- Global Rate Limiting
    - lua-resty-global-throttle
    - memcached

Global Rate limits - ingress-nginx



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ingress-nginx-controller
  namespace: ingress-nginx
data:
  global-rate-limit-memcached-host: memcached.nginx-ingress.svc.cluster.local
  global-rate-limit-memcached-port: 11211
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/#global-rate-limit

Global Rate limits - ingress-nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-example
  annotations:
    nginx.ingress.kubernetes.io/global-rate-limit: 100
    nginx.ingress.kubernetes.io/global-rate-limit-window: 5s
    nginx.ingress.kubernetes.io/global-rate-limit-key: "${remote_addr}-${http_x_api_client}"
    nginx.ingress.kubernetes.io/global-rate-limit-ignored-cidrs: "8.8.8.8"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/#global-rate-limit

Global Rate limits - ingress-nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-example
  annotations:
    nginx.ingress.kubernetes.io/global-rate-limit: 100
    nginx.ingress.kubernetes.io/global-rate-limit-window: 5s
    nginx.ingress.kubernetes.io/global-rate-limit-key: "${remote_addr}-${http_x_api_client}"
    nginx.ingress.kubernetes.io/global-rate-limit-ignored-cidrs: "8.8.8.8"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting

Global Rate limits - ingress-nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-example
  annotations:
    nginx.ingress.kubernetes.io/global-rate-limit: 100
    nginx.ingress.kubernetes.io/global-rate-limit-window: 5s
    nginx.ingress.kubernetes.io/global-rate-limit-key: "${remote_addr}-${http_x_api_client}"
    nginx.ingress.kubernetes.io/global-rate-limit-ignored-cidrs: "8.8.8.8"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting

Global Rate limits - ingress-nginx



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-configuration-example
  annotations:
    nginx.ingress.kubernetes.io/global-rate-limit: 100
    nginx.ingress.kubernetes.io/global-rate-limit-window: 5s
    nginx.ingress.kubernetes.io/global-rate-limit-key: "${remote_addr}-${http_x_api_client}"
    nginx.ingress.kubernetes.io/global-rate-limit-ignored-cidrs: "8.8.8.8"
spec:
  ingressClassName: nginx
  rules:
  - host: custom.configuration.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: http-svc
            port: 8080
```

https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting

Rate limits - service mesh



Istio

Rate limits - service mesh



CLIENT1

Client APP

Sidecar proxy

CLIENT2

Client APP

Sidecar proxy

service

POD1

Nginx

php-fpm

Sidecar proxy

POD2

Nginx

php-fpm

Sidecar proxy

Rate limits - istio - connectionPool



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-destinationrule
  namespace: prod
spec:
  host: microservice
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        http2MaxRequests: 200
        http1MaxPendingRequests: 5
```

https://istio.io/latest/docs/reference/config/networking/destination-rule/#ConnectionPoolSettings

Rate limits - istio - connectionPool



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-destinationrule
  namespace: prod
spec:
  host: microservice
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        http2MaxRequests: 200
        http1MaxPendingRequests: 5
```

https://istio.io/latest/docs/reference/config/networking/destination-rule/#ConnectionPoolSettings

Rate limits - istio - connectionPool



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-destinationrule
  namespace: prod
spec:
  host: microservice
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        http2MaxRequests: 200
        http1MaxPendingRequests: 5
```

https://istio.io/latest/docs/reference/config/networking/destination-rule/#ConnectionPoolSettings

Rate limits - istio - connectionPool



```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: microservice-destinationrule
  namespace: prod
spec:
  host: microservice
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
      http:
        http2MaxRequests: 200
        http1MaxPendingRequests: 5
```

https://istio.io/latest/docs/reference/config/networking/destination-rule/#ConnectionPoolSettings

Rate limits - service mesh - connectionPool



CLIENT1
Client APP
Sidecar proxy

CLIENT2
Client APP
Sidecar proxy

service

POD1
Nginx
php-fpm
Sidecar proxy

POD2
Nginx
php-fpm
Sidecar proxy

Rate limits - istio - rate limits - envoy



- Local rate limit
- Global Rate Limiting
  - gRPC
  - Redis

https://istio.io/latest/docs/tasks/policy-enforcement/rate-limit/

Rate limits - istio - rate limits - envoy



- Local rate limit
- Global Rate Limiting
  - gRPC
  - Redis

EnvoyFilter == configuration snippet

Rate limits - istio - rate limits - envoy



- Local rate limit
- Global Rate Limiting
  - gRPC
  - Redis

EnvoyFilter == configuration snippet

TCP/HTTP
SOURCE/DESTINATION
INBOUND/OUTBOUND

Rate limits - istio - local rate limits - envoy



- Local rate limit
- Global Rate Limiting
  - gRPC
  - Redis

token bucket:

max_tokens: количество запросов в обработке
tokens_per_fill: количество новых запросов
fill_interval: окно приёма запросов

https://istio.io/latest/docs/tasks/policy-enforcement/rate-limit/#local-rate-limit

Rate limits - istio - global rate limits - envoy



- Local rate limit
- Global Rate Limiting
  - gRPC
  - Redis

global rate limiting service

rate_limit:
  unit: время
  rate_limit: количество запросов

https://istio.io/latest/docs/tasks/policy-enforcement/rate-limit/#global-rate-limit

Rate limits - API Gateway







Rate limits - API Gateway





In-depth feature comparisons for Gravitee vs Kong API Management

API Gateway and API Management console

Features	Gravitee	Kong
UI available in addition to a Gateway	✅	⚠️
Owns the entire technology stack	✅	🚫
Service-mesh specific capabilities built into APIM	🚫	✅
Advanced Kubernetes operator	✅	✅
Supports CI/CD use cases	✅	✅
REST API support	✅	✅
SOAP support	✅	✅
GraphQL support	⚠️	⚠️
Kafka support	✅	⚠️
gRPC support	✅	⚠️
Websocket support	✅	✅
Webhooks support	✅	🚫
Protocol mediation	✅	⚠️
Event-native API Management	✅	🚫
No-code, no-XML policy configuration	✅	🚫
API Developer Portal included	✅	⚠️
Promote APIs across environments	✅	🚫



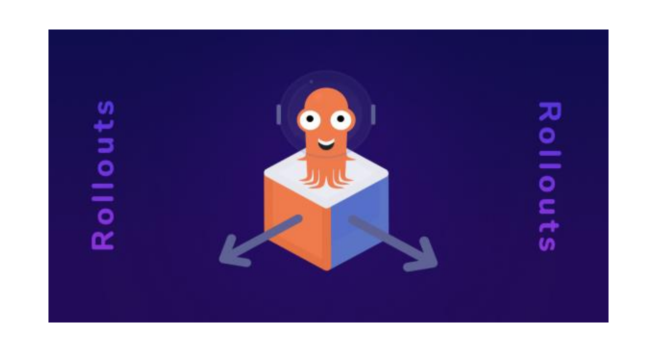
Паттерны отказоустойчивости



- Watchdog
- Health check
- Retry
- Timeouts/Deadlines
- Circuit breaker
- Rate limits
- Rollout



Canary vs Blue-Green vs a/b vs Rolling deployment







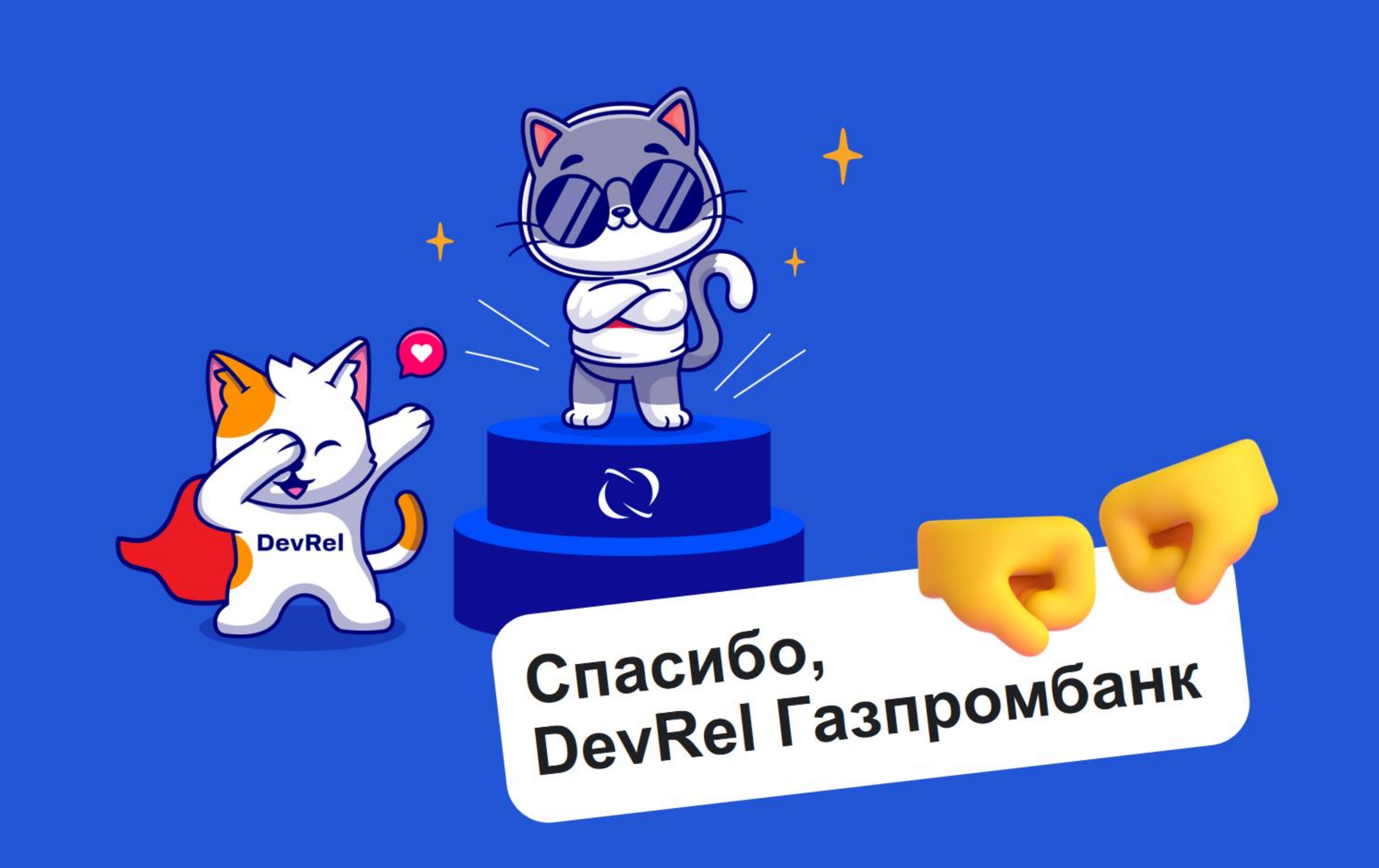
Rollout



Canary vs Blue-Green vs a/b vs Rolling deployment



Голосуйте за доклад!


DevRel
Спасибо,
DevRel Газпромбанк

Благодарности



Адель Макашева
8 (996) 469-73-40
adel.makasheva@gazprombank.ru
@adele_makasheva

Как довести спикера
до первого места на Хайлоад
и при этом не довести до ручки

https://youtu.be/B2ypGCOe4n4

Благодарности



ФЛАНТ

Yandex Cloud

Istio

рекогносцируем перед сервисмешиванием

Андрей Половов

https://youtu.be/9CUfaeT3T-A



Вопросы?



Голосуйте за доклад!



Презентация

Олег Вознесенский

Руководитель разработки отдела развития инфраструктуры для анализа данных

